SPACE VARYING GAMUT MAPPING

Technical Field

The technical field is color image processing using gamut correction.

Background

Gamut mapping is a method used to modify a representation of a color image to fit into a constrained color space of a given rendering medium. A laser-jet color printer that attempts to reproduce a color image on regular paper would have to map the photographed picture colors in a given color range, also known as the image "color gamut," into the given printer/page color gamut. Conventional gamut mapping methods involve a pixel by pixel mapping (usually a pre-defined look-up table) and ignore the spatial color configuration. More recently, spatial dependent approaches were proposed for gamut mapping. However, these solutions are either based on heurestic assumptions or involve a high computational cost. An example of such a method is discussed in "Color Gamut Mapping Based on a Perceptual Image Difference Measure," S. Nakauchi, et al., Color Research and Application, Vol. 24, pp. 280-291 (1999).

Another method relies on preserving the magnitude of gradients in an original image, while projecting the gradients onto the target gamut as a constraint. This multi-scale property is achieved by sampling the image around each pixel with exponentially increasing sampling intervals while the sampling is done along vertical and horizontal axes. The method involves modulating an L_2 measure for image difference by *human contrast sensitivity* functions. The method uses a model in which the contrast sensitivity function is a linear combination of three spatial band-pass filters H_1, H_2, H_3, given in the spatial-frequency domain (or h_1, h_2, h_3, as their corresponding spatial filters), as shown in Figure 1.

For gamut mapping of the image u_0 in the CIELAB (which stands for Commission International de I'Eclairage (CIE) with L, A, and B (more commonly L, a, b) respectively representing converted signals for cyan (C), magenta (M) and yellow (Y)) space, the method minimizes the functional

$$E\left(u^L, u^a, u^b\right) = \sum_{i=1}^{3} \sum_{c \in \{L,a,b\}} \int_{\Omega} \left(h_i^c * \left(u^c - u_0^c\right)\right)^2 d\Omega, \qquad (1)$$

subject to $\{u^L, u^e, u^b\} \in \vartheta$.

In Equation (1), h_i^c is the filter corresponding to the spectral channel $c \in \{L, a, b\}$ the i $\in \{1, 2, 3\}$ 'contrast sensitivity' mod, Ω is the image domain, and ϑ is the target gamut. Note that a total of nine filters H_i^c are involved, three for each spectral channel and a total of three spectral channels.

The filters H_i^c are modeled by shifted Gaussians. H_i^c is not shifted, and thus, h_i^c is also Gaussian, while h_2^c and h_3^c are Gaussians modulated by two sine functions with different periods. A graphical analysis of h_2^c and h_3^c as shown in Figure 2 reveals that h_2^c and h_3^c approximate the derivative operator at different scales. These two gradient approximation operators are denoted by $\nabla_{\sigma_1}^c$ and $\nabla_{\sigma_2}^c$. Note that any band pass filter can be considered as a version of a derivative operator. Furthermore, one possible extension of the 1D derivative to 2D is the gradient. Thus, the minimization of the Equation (1) functional is similar to minimizing the following functional for each channel separately.

$$\iint_{\Omega} \left|h_1^c * (u - u_0)\right|^2 + \left|\nabla_{\sigma 1}^c (u - u_0)\right|^2 + \left|\nabla_{\sigma 2}^c (u - u_0)\right|^2 d\Omega \qquad (2)$$

Roughly speaking, the first term corresponds to the S-CIELAB perceptual measure, while the next two terms capture the need for matching the image variations at two selected scales that were determined by human perception models. One technical difficulty of the spatial filters corresponding to Equation (1) is their large numerical support.

Another simple spatial-spectral measure for human color perception was proposed in "A Spatial Extension of CIELAB for Digital Color Image Reproduction," Zhang et al., Proceedings of the SID Symposium, Vol. 27, pp. 731-34, (1996). The 'S-CIELAB' defines a spatial-spectral measure for human color perception by a composition of spatial band-pass linear filters in the opponent color space followed by the CIELAB Euclidean perceptual color.

Summary

Gamut mapping is a method to modify representation of a color image to fit into a constrained color space of a given rendering medium. A laser-jet color printer that attempts to reproduce a color image on regular paper would have to map the photographed picture colors in a given color range, also known as the image "color gamut," into the given printer/page color gamut.

A method an apparatus uses a variational approach for space dependent gamut mapping. The method presents a new measure for the problem. A solution to the formulation according to the method is unique if the gamut of the target device is known. A quadratic programming efficient numerical solution provides real-time results.

In an embodiment, the method comprises receiving an input image, converting color representations of an image pixel set to produce a corresponding electrical values set, applying a space varying algorithm to the electrical values set to produce a color-mapped value set, and reconverting the color-mapped value set to an output image. The space varying algorithm minimizes a variational problem represented by

$E(u)=\int_{\Omega}\left(D^2+\alpha\left|\nabla D\right|^2\right)d\Omega x$, subject to $u \in \vartheta$, where Ω is a support of the image, α is a non-negative real number, $D=g*(u-u_0)$, and g is a normalized Gaussian kernel with zero mean and a small variance σ.

The method may be implemented in a variety of image capture reproduction devices, including cameras and printers. The method may be embodied on a computer-readable medium, such as a CD-ROM, for example.

Description of the Drawings

The detailed description will refer to the following drawings, in which like numerals refer to like objects, and in which:

Figure 1 is a qualitative description of filters modeling the human contrast sensitivity functions in the spectral domain;

Figure 2 illustrates a shifted Gaussian;

Figure 3 is a block diagram of an apparatus that uses a variational approach for gamut mapping;

Figures 4-7 illustrate algorithms and processes used with the apparatus of Figure 3;

Figure 8 illustrates an example of a behavior of an algorithm used by the apparatus of Figure 3;

Figure 9 is a flowchart of an algorithm illustrating an alternative operation of the apparatus of Figure 3; and

Figure 10 is a block diagram of a computer system that implements the algorithm of Figure 4.

Detailed Description

Gamut mapping is a method used to modify a representation of a color image to fit into a constrained color space of a given rendering medium. A laser-jet color printer that attempts to reproduce a color image on regular paper would have to map the photographed picture colors in a given color range, also known as the image "color gamut," into the given printer/page color gamut. Conventional gamut mapping methods involve a pixel by pixel mapping (usually a pre-defined look-up table) and ignore the spatial color configuration. More recently, spatial dependent approaches were proposed for gamut mapping. However, these solutions are either based on heurestic assumptions or involve a high computational cost.

The gamut mapping problem is related to the Retinex problem of illumination compensation and dynamic range compression. The basic Retinex problem is: How to estimate the reflectance image from the given acquired image? A reasonable optical model of the acquired image *S* asserts that it is a multiplication of the reflectance *R* and the illumination *L* images. Where the reflectance image is a hypothetic image that would have been measured if every visible surface had been illuminated by a unit valued white illumination source, and the illumination image is the actual illumination shaded on surfaces in the scene. In the log domain:

$$s = r + l$$

where s, r, and l are the respective logarithms of S, R, and L. Since surface patches can not reflect more light than has been shaded on them, $R < 1 \Rightarrow r < 0$. Thus, in an ideal situation, image $r < 0$, which is perceptually similar to s. For the Retinex, an additional physically motivated constraint is provided, namely, that the illumination image $l = s - r$ is smooth, i.e. the gradient $|\nabla l| = |\nabla(r - s)|$ is small. But this is just another way to say that the features of r are similar to those of s, since the illumination is assumed not to create perceptual features in s. In the gamut mapping problem an image u_0 exists, and the problem is to estimate an image $u \in \vartheta$, which is not only perceptually similar to u_0, but also has similar perceptual features as u_0.

A good measure of image deviation captures the perceptual difference between the initial, u_0 and the final, u, images. This is modeled by

$$D = g * (u - u_0). \tag{3}$$

where g may be a normalized Gaussian kernel with zero mean and a small variance σ. This model is good for small deviations. However, for large deviations it should be elaborated to account for possible perceptual *feature* differences, which may be modeled by the difference of gradients, which due to linearity, turns out to be the gradient of Equation (3)

$$\nabla D = \nabla[g * (u - u_0)] = g * (\nabla u - \nabla u_0) \tag{4}$$

The proposed measure yields the functional

$$E(u) = \int_{\Omega} \left(D^2 + \alpha |\nabla D|^2\right) d\Omega, \tag{5}$$

subject to $u \in \vartheta$,

which should be minimized.

Taking a first variation of Equation (5) with respect to u yields the Euler-Lagrange equation:

$$\frac{\delta \mathrm{E}(u)}{\delta u} = g * (\alpha div(\nabla g * (u - u_0)) - g * (u - u_0)) = 0, \tag{6}$$

Reformulating Equation (6) as a gradient descent flow for u, provides the following minimization scheme:

$$\begin{cases} \dfrac{du}{dt} & = & \alpha g * \Delta D - g * D \\ s.t. & & u \in \vartheta \end{cases} \tag{7}$$

The functional (Equation (5)) and the resulting minimization scheme are both Euclidean invariant in the image plane. They are thus both translation and rotation invariant. As the parameter α goes to zero, the S-CIELAB model is approximated, while for effective α the result is a proper extension to the perceptual measures, with an efficient numerical implementation.

To provide a numerical solution, recall that an artificial time parameter t was added to the image $u(x, y)$, which now reads $u(x, y, t)$. A first step is to discretize the Eueler-Lagrange gradient descent equation, by first taking a simple forward explicit approximation for the t derivative,

$$\frac{u^{n+1} - u^n}{\tau} = \alpha g * \Delta D - g * D$$

where $\tau = dt$ and $u^n(x, y) \approx u(x, y; n\tau)$.

Next, the space derivatives are solved. Let $u_{ij}^n \approx u(ih, jh, nt)$, where uniform spatial spacings are assumed in the x and y directions of size h. Using central derivatives in space,

$$\begin{array}{lcccc} u_{xx} & \approx & D_{xx}u & \equiv & \dfrac{u_{i+1} - 2u_i + u_{i-1}}{h^2} \\ u_x & \approx & D_x u & \equiv & \dfrac{u_{i+1} - u_{i-1}}{2h} \end{array}$$

and the same for the y direction. Using the relation $g * D_{xx} (g * u) = g_x * g_x * u$, the algorithm next computes the kernels $D_2 = g_x * g_x + g_y * g_y = D_x g * D_x g + D_y g * D_y g$. The explicit approximation reads:

$$\begin{array}{lcl} \overline{D}^n & = & g * g * \left(u^n - u_0\right) \\ L^n & = & D_2 * \left(u^n - u_0\right) \\ u_{ij}^{n+1} & = & u_{ij}^n + \tau\left(\alpha L_{ij}^n - \overline{D}_{ij}^n\right) \end{array}$$

subject to the constraint $u_{ij}^n \in \vartheta$.

To speed up convergence, a standard coarse to fine pyramidal approach may be preferred. For example, an original image S may be composed of a set of 512 x 512 pixels. The image S may then be averaged down (sub-sampled or decimated) by taking, for example, every 4 adjacent pixels and replacing the block of four pixels with a single pixel having a value that represents an average of the four pixels. This results in an image set that is 256 x 256 pixels. The sub-sampling can be continued, yielding a subsequent reduced pixel set, or layer, comprising 128 x 128 pixels. This process may continue, with each resulting layer being an increasingly coarser version of the original image S. The final layer will be a set of only 2 x 2 pixels, and likely will be blurred. The process of sub-sampling or decimating to provide increasingly coarser details results in an image pyramid. The original image *S* represent the base of the image pyramid at 512 x 512 pixels, followed by layers of 256 x 256 pixels, 128 x 128 pixels, 64 x 64 pixels, and so on. Using this approach of an image pyramid allows for rapid convergence of the solution to the functional of Equation (5). The process starts by using the coarsest layer of the image pyramid, and running a number of iterations until a solution is achieved. The solution is then used to initialize the next finer layer of the image pyramid, resulting in less computation in order to arrive at a solution. The process continues until the finest layer (e.g., the original, full resolution, image set for *S*) is reached. As an alternative to the image pyramid, a full multi-grid method may be used.

The functional of Equation (5) has a Quadratic Programming (QP) form, since the penalty term is quadratic and the constraint is linear. If the set ϑ is convex, the overall problem is convex if and only if the Hessian of the functional Equation (5) is positive definite. In such a case, there is a unique local minimum that is also the global solution to the problem. In the above embodiment, the Hessian is given by $g * (1 - \alpha\Delta) * g$, which is positive definite for all $\alpha > 0$. Thus, for a convex target gamut ϑ, there exists a unique solution.

The above-described functional (Equation (5)) may be used in imaging devices to map a gamut of an image to the gamut of a device that attempts to reproduce the image. Figure 3 is a block diagram of an apparatus 100 that may be used for gamut mapping according to the functional of Equation (5). An image capture device 101 receives an input original image and produces an output image *S* 102, which is an electrical signal representing a colorimetric value image. For example, the capture device 101 may convert at least three colorimetric values of each pixel of the original image into corresponding electrical signals. The electrical signals may indicate the L, a, b values, for example. Other colorimetric values may be the XYZ tristimilus value and the L, U, V or device dependent RGB values. A gamut mapper 105 produces an in-gamut image S′ 106. Finally, a rendering module 107 provides a rendered image *S″* 108. The rendering module 107 may be implemented as a color laser printer. The thus-generated image *S″* 108 may represent a best-fit image, given gamut limitations of the device in which the image reproduction is to occur.

The image *S* 102 may represent the image as sensed by the capture module 103. The gamut mapper 105 applies an algorithm to extract and map the values of the image *S* 102 into the gamut of the image reproduction device 107. In particular, the gamut mapper 105 may apply an algorithm that solves the problem represented by Equation (5), thereby solving the gamut mapping problem and optimizing the output image S'' 108.

Figure 4 is a block diagram showing routines of an algorithm 120 that may be used to complete the gamut mapping function. In an embodiment, the algorithm 120 provides a numerical solution to the functional of Equation (5).

The algorithm 120 begins with input block 121, where an input to the algorithm 120 is an image S of size $[N, M]$, and the two parameters α and β.

In initialization block 123, a Gaussian pyramid of the image s is computed. The thus-constructed pyramid contains p resolution layers (from fine (1) to coarse (p)) with the current resolution layer (k) set to p. In block 125, T_k iterations of a gradient descent algorithm are applied to the k^{th} resolution layer, until all resolution layers are checked, block 127. In block 129, the next resolution layer is updated. When all resolution layers are processed, the result is the final output of the algorithm 120.

Figure 5 is a block diagram showing the initialization routine 123 in more detail. In block 131, a counter is initialized with i = 1. In block 132, the Gaussian pyramid is constructed by smoothing the image with a kernel, such as the kernel k_{PYR}:

$$K_{PYR} = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$

In block 133, the image is decimated by, for example, a 2:1 ratio. The process is repeated (block 134) and the counter is incremented (block 135) p times where $p < \lg_2(\min(M, N))$. This process produces a sequence of images $\{S_k\}_{k=1}^{p}$ conventionally named the "Gaussian Pyramid of S." The image S_1 is the original image S, and S_p is an image with the coarsest resolution for the Gaussian pyramid.

In block 137, the process sets $k = p$, i.e., starts at the coarsest resolution layer k, and sets the initial condition $L_0 = \max\{S_p\}$.

Figure 6 is a block diagram of an embodiment of the processing main routine 125. The routine 125 starts at block 141, where the new resolution layer is initialized and T_k and α_k are set (e.g., $T_k = K{*}T_0$).

Then, for $j = 1, .., T_k$.

In block 143, the routine 125 calculates the gradient:

$$G = \Delta(u - u_0) + \alpha_k(u - u_0)$$

where Δx is the convolution of each of the color planes of x with K_{LAP}:

$$K_{LAP} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ and } \alpha_k \text{ is, for example } \alpha_k = \alpha * 2^{2(k-1)}$$

In block 145, the routine 125 calculates μ_{NSD}:

$$\mu_{NSD} = \Sigma G^2 \Big/ \left(\Sigma(G * \Delta G) + \alpha_K \Sigma G^2\right)$$

In block 147, the routine 125 completes the gradient descent iteration:

$$L_j = L_{j-1} - \mu_o \cdot \mu_{NSD} \cdot G,$$

Where μ_o is a constant; for example μ_o=0.8.

In block 148, the result is projected onto the constraints:

$$L_j = \text{Proj}_\vartheta(L_j)$$

Where $\text{Proj}_\vartheta(x)$ is a projection of x into the gamut ϑ.

In block 149, if $j \neq T_K$, processing returns to block 143. Otherwise, the routine 125 ends.

Returning to Figure 4, in block 127, if $k > 1$, the result L_{Tk} is up scaled (2:1 ratio) by pixel replication into the new L_0, the initialization for the next resolution k - 1 layer. The resolution layer is updated $k = k - 1$, and the algorithm proceeds by going again to block 125. If $k = 1$, the result L_{T1}, is the final output of the algorithm 120.

Figure 7 is a block diagram of an alternative embodiment of an algorithm routine, denoted as 120a. The routine 120a may be applied to solve the equation:

$$u_{ij}^{n+1} = u_{ij}^n + \tau\left(\alpha L_{ij}^n - \overline{D_{ij}^n}\right)$$

subject to the constraint $u_{ij}^n \in \vartheta$.

The routine 120a begins with block 151, where the kernels D_2 and g, and the counter k are initialized.

In block 153, the routine 120a calculates:

$\overline{D^n} = g * g * (u^n - u_0)$.

In block 155, the routine 120a calculates:

$L^n = D_2 * (u^n - u_0)$.

In block 157, the routine 120a performs the gradient steps to solve u_{ij}^{n+1} as noted above.

In block 158, the routine 120a determines if k>1, and if so, processing moves to block 159, where k is set to k-1. Processing then returns to block 153. Otherwise, the routine 120a ends.

The penalty function as shown in Equation (5) tends to create halos in the resulting image. Figure 8 explains the origin of those halos through a one dimensional example. Figure 8 shows an original signal 160 that is outside of gamut 162 (delimited in the gray range by dotted lines 163 and 164). Projecting the signal 160 onto the gamut 162 will result in a constant value, denoted as "High" (163) and loss of all detail. Figure 8 also shows the result of processing the original signal 160. Dashed line 165 represents the result of scaling the signal 160 into the allowed range of the gamut 162. All the details are preserved, but with a smaller contrast. As opposed to point operations, the space dependent approach represented by Equation (5) yields a signal 166 (the solid line) that preserves the details with high contrast. However, halos occur near strong edges 167, which means that near the edges 167 there is a slow transition from low to high values.

In order to avoid these phenomena, the penalty term may be modified, and robust estimation tools may be used. The original penalty term in Equation (5) may be replaced by:

$$E(u) = \int_\Omega \left(\rho_1(D) + \alpha\rho_2(|\nabla D|)\right) d\Omega. \quad (8)$$

which for $\rho_1(x) = \rho_2(x) = x^2$ coincides with Equation (5). If the function $\rho(x)$ grows slower than x^2 as $x \rightarrow \infty$, behavior near strong edges improves. Good candidates for $\rho(x)$ are $\rho(x) = |x|$ or $\rho(x) = \sqrt{1+x^2}$.

A different and simpler (linear) approach with similar robust behavior involves solving the original Equation (5) twice, with two different values of α. A solution with a small α may be denoted as u_{small} and a solution that corresponds to the high value of α may be denoted as u_{high}. The solution u_{small} has smaller contrast at areas with small details, yet has almost no halos. On the other hand, the solution u_{high} preserves the small details, but at the expense of strong halo effects. By averaging these two results (u_{small} and u_{high}) in a spatially adaptive way provides a simple, improved solution. The improved solution is therefore:

$$u_{final}[k,j] = w[k,j]u_{small}[k,j](1 - w[k,j])u_{high}[k,j]$$

The weight $w[k, j]$ should be close to one near strong edges, and close to zero in relatively smooth regions. In an embodiment,

$$w[k,j] = \frac{1}{1 + \beta|\nabla g * u_0|^2}$$

provides a robust estimation.

Halo problems have been recently dealt with in relation to Dynamic Range Compression. Solutions proposed included anisotropic diffusion and robust filtering. The halo-related solutions described herein are solutions to the same halo problem.

Figure 9 is a flowchart showing an operation 180 of the gamut mapper 105. The process begins in start block 185. In converter block 195, the raw image S is converted from electrical signals having arbitrary values (colorimetric values) to locally defined color descriptors.

In block 200, the converted image signal is decimated to form an image pyramid. A 2:1 decimation scheme may be used, for example. Subsequent steps in the operation 180 begin with the coarsest resolution layer of the image pyramid using a space varying algorithm. In block 205, the gamut mapper 105, using a space varying algorithm such as that represented by Equation (5) calculates the image deviation for the coarsest resolution layer. This process may involve calculating a first variation (block 210), determining a gradient descent flow (block 215), and solving the resulting gradient subject to a constraint (block 220). In block 225, a determination is made if the current resolution layer is the last (finest) resolution layer. If additional layers remain for processing, the process 180 returns to block 205. Otherwise, the process 180 moves to block 230, where local color descriptors are converted to the required output format. In block 245 the process 180 ends.

The above-described space-varying algorithm 120, as represented by Equation (5), for example may be executed using a general purpose computer system 250 as shown in Figure 10. The computer system 250 includes an input device 260, a computer unit 270, a display 280, a keyboard 290, and a storage device 300. The storage device 300 may be a CD-ROM drive, for example. Program code instructions for executing the algorithms 120, 120a and 180 may be stored on a CD-ROM 305, or other computer readable memory, which is read by the storage device 300. The program code read out of the CD-ROM 305 are executed by a CPU of the computer unit 270. The CPU may perform the processing shown by the flowchart of Figure 9. The algorithms 120, 120a, and 180 also may be performed in a camera or printer hardware.